MASTERMIND PROTOCOL -- PEX 1

**1 Introduction**

This protocol was developed to successfully play the game Mastermind over a network using UDP. Mastermind is a game in which the player has 10 attempts to guess a correct pattern of 4 random letters. The pattern can contain any combination of the letters a-f. This protocol allows for easy communication between the client and server Mastermind applications.

**2 Overall Operation**

Clients must attempt to connect to the server AFTER the server has been started. The server will generate an instance of the game Mastermind and then wait for UDP requests from the client. It will only respond to requests; it will not initiate communication with the client.

**3 Message Format**

The client and server use the same message format that consists of a single string. This string is in a comma separated value format. When either the client or server receives a datagram, the data is split into an array by the commas in the string. There are no spaces between each comma and string.

Sent Data: String data = “guess,abcd”;

Parsed Data: String[] data = {“guess”, “abcd”};

The first value in the string is always the action identifier and ends up being the first value in the parsed array. The client and server will use the action identifier to determine how to handle the resulting data array. All of the remaining values in the string and parsed array are data values. There can be any number of data values to include none. Commands that do not have data values do not need a comma after the action identifier.

Sent Data: String data = “state”;

Parsed Data: String[] data = {“state”};

When a datagram is received the action identifier is matched to a set of commands to be executed. These commands expect a certain number of data values in a certain order to be successfully executed.

EXAMPLE:

For every client connection the server always receives the following request:

String data = “state”;

Which the server parses to:

String[] data = {“state”};

The server then responds with:

String data = “state,0,playing,1,4,deca”;

The client receives this and parses it to:

String[] data = {“state”, “0”, “playing”, “1”, “4”, “deca”};

The client finds the commands to execute based on the action identifier of “state” and knows what each data value represents and uses them accordingly.

In this case the data values are # correct, game state, guess #, puzzle size, and pattern.

Since the client and server know what data values to expect for each action identifier, issues like type casting are handled by the respective action identifier commands.

**4 Commands/Messages**

The set of commands for the Mastermind protocol are defined below. This set should define each of the command/message types that are transmitted between the client and server.

4.1 COMMAND #1

“state” – Sent by the client and causes the server to respond with the state of the current mastermind game.

“state,# correct,game state,guess #,puzzle size,pattern” – The formatting of the servers response to the request of state. The client will display this information to the user as appropriate.

4.2 COMMAND #2

“guess,user guess” – Send by the client and causes the server to respond with how well the guess matched the pattern.

“guess,result string,game state,guess #,puzzle size,pattern” – The servers response to the client. The client will display this information to the user as appropriate and possibly end the game if required.

4.3 COMMAND #3

“answer” – Sent by the client and causes the server to send back the pattern.

“answer,pattern” – The response from the server. The client will display this information to the user so they know how close their guess was.

4.4 COMMAND #4

“numtries” – Sent by the client and causes the server to send back the number of guesses made.

“numtries,tries” – The response from the server. The client will display this information to the user so they know how many guesses have been used.

4.5 COMMAND #5

“reset” – Sent by the client and causes the server to reset its current mastermind game to a new puzzle.

“reset” – The response from the server. The client will inform the user a new pattern has been generated.

4.6 COMMAND #6

“history” – Sent by the client and causes the server to send back its current history of guesses.

“history,history data” – The response from the server. The client will display the history of guesses or “No History” if no history exists.

4.7 COMMAND #7

“set,pattern” – Sent by the client and causes the server to set its pattern to the specified pattern.

“set,pattern” – The response from the server. The client will inform the user their pattern was set.

4.8 COMMAND #8

“error,message” – The response from the server in the event it receives a command identifier it does not recognize. This will cause the client to terminate operation before it can break anything else.

**5 Author**

C2C Gavin Delphia

Email: C14Gavin.Delphia@USAFA.edu